



HAI811I Programmation mobile

Rendu Intermédiaire 2

Projet Mobile StageConnect

Réalisé par :

BELLIL MOHAMED NADIR

GUERFI MOHAMED

Table des matières

Introduction.....	3
Outils et technologies utilisés.....	3
Frontend mobile.....	3
Backend.....	3
Communication.....	4
Description de l'architecture de l'application.....	4
Présentation (Presentation Layer).....	4
Domaine (Domain Layer).....	4
Données (Data Layer).....	4
Backend.....	5
Notifications et Messagerie.....	5
Sécurité.....	5
Schéma de l'architecture.....	6
Architecture du code.....	7
Data - Couche Data Layer.....	8
Domain - Couche Domaine / Métier.....	8
Presentation - Couche UI / Présentation.....	9
ui/theme - Couleurs et styles globaux.....	9
Fichiers Root.....	9
Relations entre couches.....	10
Exemple de flux d'inscription.....	10
Architecture du code (back-end).....	11
Packages du code (com.example.stageconnect).....	12
resources.....	13
Fonctionnalités Réalisées.....	13
Gestion d'Authentification et d'Inscription.....	13
Gestion du Profil Utilisateur.....	13
Gestion des Fichiers.....	13
Consultation et Filtrage des Offres.....	14
Interfaces Réalisées.....	15
Avancement et prochaines étapes.....	28
Ce qui est réalisé.....	28
Ce qui reste à faire.....	28
Modules complémentaires.....	28

Introduction

Le projet **StageConnect** vise à développer une application mobile de gestion des stages, centralisant la publication d'offres, la recherche et la candidature, la validation académique, ainsi que la communication entre les étudiants et les entreprises ainsi que les établissements de formation. Ce rapport intermédiaire présente l'architecture de l'application, les outils et technologies utilisés, ainsi qu'un état d'avancement du projet.

Outils et technologies utilisés

Frontend mobile

1. **Kotlin** avec **Jetpack Compose** pour l'interface utilisateur moderne et réactive.
2. **Dagger Hilt** un outil d'injection de dépendances pour Android, basé sur Dagger. Hilt automatise la gestion du cycle de vie des objets, facilite le découplage des composants et améliore la testabilité de l'application. Il est particulièrement adapté à l'architecture modulaire et favorise la réutilisation du code.
3. **Clean Architecture** : Approche architecturale qui sépare l'application en plusieurs couches indépendantes
4. **Présentation** : gestion de l'interface utilisateur et des interactions.
5. **Domaine** : logique métier et cas d'utilisation.
6. **Données** : sources de données (API, base locale, etc.).

Backend

1. **Spring Boot** pour la création d'API REST robustes.
2. **Spring Security** module de sécurité de Spring pour la gestion de l'authentification et des autorisations. Il permet de protéger les endpoints de l'API, de gérer les rôles utilisateurs (étudiant, entreprise, université) et d'assurer la confidentialité et l'intégrité des données échangées.
3. **JWT (JSON Web Token)** pour l'authentification sécurisée entre le mobile et le backend.
4. **Validators** ensemble de mécanismes de validation des données reçues (formulaires, requêtes API, etc.), permettant de garantir la cohérence, la sécurité et la fiabilité des opérations côté serveur.

5. **PostgreSQL** comme base de données relationnelle pour la persistance des données.

Communication

- **API REST** pour l'échange de données entre l'application mobile et le backend.

Description de l'architecture de l'application

L'architecture suit les principes de la **Clean Architecture**, adaptée aux applications mobiles modernes, et se compose des couches suivantes :

Présentation (Presentation Layer)

- Gérée par Jetpack Compose : écrans, navigation, gestion des états UI.
- Interaction avec la couche Domain via des ViewModels injectés par Dagger Hilt.
- Responsabilités : affichage des offres, formulaires de candidature, messagerie, notifications, etc.

Domaine (Domain Layer)

- Contient la logique métier indépendante de l'implémentation technique.
- Définit les cas d'utilisation (UseCases) : publier une offre, postuler à un stage, valider un stage, etc.
- Interfaces des repositories pour abstraction de la source de données.

Données (Data Layer)

- Implémentation concrète des repositories.
- Gestion de la communication avec le backend via des services réseau (API REST).
- Mapping des modèles de données entre le format réseau, la base locale et le domaine

Backend

- **Spring Boot** expose des endpoints REST sécurisés par **Spring Security** et JWT.
- **PostgreSQL** stocke les utilisateurs, offres de stage, candidatures, messages, notifications, etc.
- **Validators** assurent la cohérence et la sécurité des données reçues.
- Gestion des rôles (étudiant, entreprise, établissement) et des droits d'accès.

Notifications et Messagerie

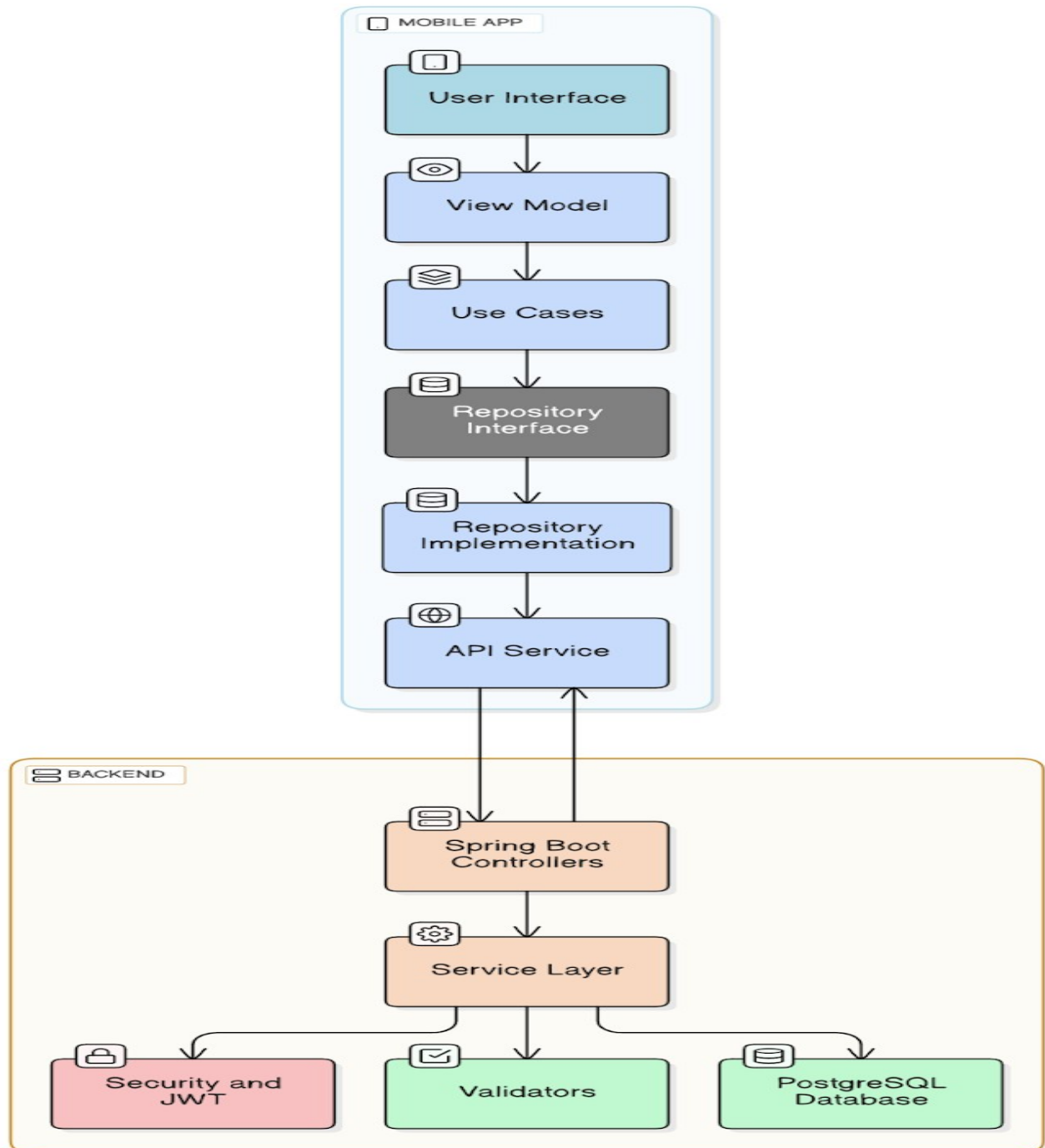
- Système de notifications internes (push ou locales) pour informer les utilisateurs des nouveaux événements.
- Module de messagerie pour les échanges entre étudiants, entreprises et établissements.

Sécurité

- Authentification par JWT : chaque requête mobile inclut un token pour vérifier l'identité et les droits de l'utilisateur.
- Gestion des autorisations selon le rôle (étudiant, entreprise, établissement).

Schéma de l'architecture

Mobile App and Backend Flow



Ce schéma illustre la séparation des responsabilités et la circulation des données entre les différentes couches de l'application mobile et le backend.

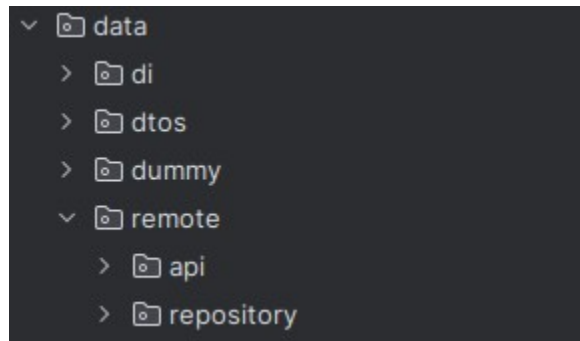
Architecture du code

```

▼ com.example.stageconnect
  > data
  ▼ domain
    > model
    > result
    > usecases
      < getFileNameAndSize.kt
  ▼ presentation
    > components
    > navigation
    ▼ screens
      > applications
      > applicationstatus
      > appstart
      > filter
      > home
      > jobdetails
      > messaging
      > onboarding
      > profile
      > savedjobs
      > search
      > signin
      > signup
      > viewmodels
    > ui.theme
  < MainActivity
  < MyApplication
```

Data - Couche Data Layer

Responsable de **recupérer les données** (API, DB, fichiers, etc.)

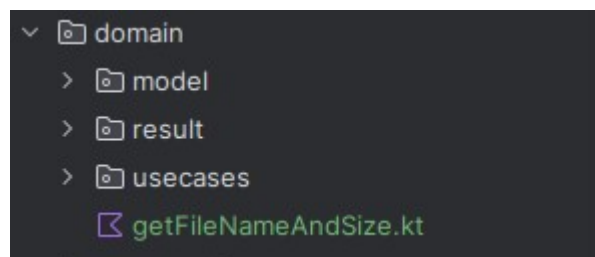


- **di/NetworkModule.kt** : Fournit les dépendances (comme Retrofit, etc.) via Hilt/Koin.
- **dtos/UserDto.kt** : Contient les **Data Transfer Objects**, utilisés pour sérialiser/désérialiser les données.
- **remote/api/ApiService.kt** : Définit les appels d'API (ex: Retrofit interface).
- **remote/repository/RegisterRepository.kt** : Implémente un **repository** qui appelle l'API ou d'autres sources.
- **dummy/** : Fournit des données fictives pour le test.

Rôle : Cette couche ne contient pas de logique métier, seulement l'accès brut aux données.

Domain - Couche Domaine / Métier

Responsable de la **logique métier pure**.

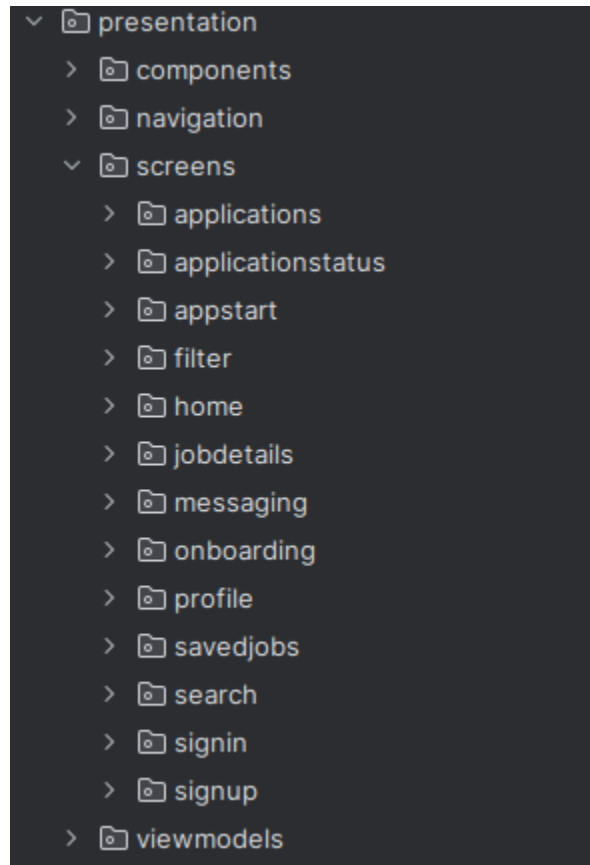


- **model/** : Contient toutes les entités métier (User, Offer, Application, etc.)
- **enums/** : Définit les types forts de ton domaine (comme JobLevel, WorkType, etc.)
- **usecases/** : Contient la logique métier spécifique (ex: RegisterUseCase, FilterUseCase)
- **result/Result.kt** : Encapsulation des résultats (succès, échec, exception...)
- **getFileNameAndSize.kt** : Utilitaire métier (lié à ton use case probablement).

Rôle : Agit comme un **pont pur** entre les données et l'UI. Aucun Android ici.

Presentation - Couche UI / Présentation

Responsable de l'**affichage**, **navigation**, **interaction** avec l'utilisateur.



- **screens/** : Toutes les pages de ton app (connexion, inscription, profil, etc.)
- **components/** : UI réutilisables (CustomEditText, AppButton, etc.)
- **viewmodels/** : Gèrent l'état de l'UI (souvent connectés aux use cases).
- **navigation/** : Définition des routes et navigation entre les écrans (AppNavHost, Screen).

Rôle : L'utilisateur interagit ici. Les ViewModels appellent les UseCases.

ui/theme - Couleurs et styles globaux

- **Color.kt**, **Theme.kt**, **Type.kt** : Définissent les couleurs, polices, thèmes généraux Jetpack Compose.

Fichiers Root

- **MainActivity.kt** : Point d'entrée principal de l'application.
- **MyApplication.kt** : Classe Application, utilisée pour initialiser Hilt/Dagger.

Relations entre couches

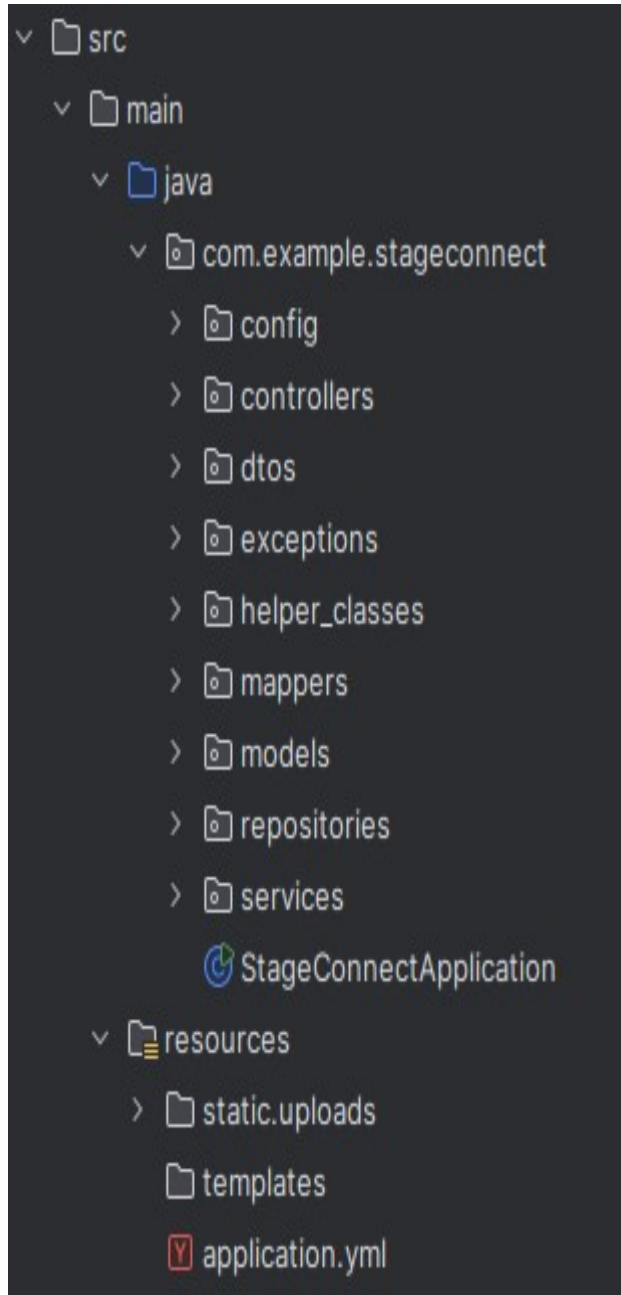
- presentation appelle les usecases du domain.
- usecases utilisent les repositories fournis par la couche data.
- data fournit les implémentations concrètes de repositories.

Exemple de flux d'inscription

1. **UI** (UserInformationScreen.kt) collecte les infos et appelle le RegisterViewModel.
2. RegisterViewModel appelle RegisterUseCase.
3. RegisterUseCase utilise RegisterRepository.
4. RegisterRepository appelle ApiService pour envoyer les données à l'API.
5. La réponse est remontée et affichée à l'utilisateur.

Architecture du code (back-end)

- **src/main/java** : Contient tout le **code source de l'application**.
- **src/main/resources** : Contient les **ressources** (fichiers de configuration, templates, fichiers statiques).
- **src/test/java** : Contient les **tests unitaires ou d'intégration**.
- **target/** : Généré par Maven ou Gradle après compilation (à ne pas modifier directement).
- **static/uploads/** : Stockage d'images ou fichiers



Packages du code (com.example.stageconnect)

Config : Configuration de sécurité et JWT

- **ApplicationConfig.java** : Fournit les beans nécessaires (comme le AuthenticationManager).
- **JwtAuthenticationFilter.java** : Filtre HTTP qui intercepte les requêtes pour extraire et valider le token JWT.
- **JwtService.java** : Gère la génération, la validation et l'extraction d'informations depuis le JWT.
- **SecurityConfig.java** : Configuration principale de Spring Security (autorisations, authentification, filtre JWT, etc.).

Controllers : Contient les **contrôleurs REST**

- **FileController.java** : Gère l'upload, le téléchargement ou l'affichage de fichiers.
- **UserController.java** : Gère les actions liées aux utilisateurs (authentification, inscription, infos, etc.).

Dtos : Data Transfer Objects – objets utilisés pour transférer les données

- **UserDto.java** : Sert à transmettre les infos de l'utilisateur sans exposer l'entité complète.

Exceptions : Gestion des erreurs

- **EmailAlreadyExistsException.java** : Exception personnalisée levée si un email est déjà utilisé.
- **GlobalExceptionHandler.java** : Gère toutes les exceptions personnalisées ou génériques via **@ControllerAdvice**.

Helper_classes : Petites classes utilitaires :

- **AuthenticationRequest.java** : Contient les données pour une requête de connexion (email, password).
- **AuthenticationResponse.java** : Réponse contenant le JWT, rôle, etc.

Mappers : Mapping entre entités et DTOs

- **Mapper.java** : Interface générique.
- **UserMapper.java** : Transforme un User en UserDto et vice versa.

Models : Les entités JPA

- User, Role, Project, WorkExperience, Language, etc. : Représentent les tables de la base de données.
Exemple : un User possède une liste d'expériences, de langues, de projets, etc.

Repositories : Interfaces pour la communication avec la base

- UserRepository.java : Étend JpaRepository<User, Long> et contient des requêtes personnalisées

Services : Couche métier (logique applicative)

- interfaces/ : Contient les **interfaces des services** (UserService, FileService) définissant les méthodes à implémenter.
- impl/ : Implémentation concrète des interfaces (UserServiceImpl, FileServiceImpl).

StageConnectApplication.java : Classe principale avec @SpringBootApplication, point d'entrée de l'application.

resources

- application.yml : Fichier de configuration Spring Boot (DB, port, JWT secrets, etc.).
- static/uploads/ : Contient les fichiers envoyés par les utilisateurs.

Fonctionnalités Réalisées

Gestion d'Authentification et d'Inscription

- Création d'un compte utilisateur avec vérification d'unicité de l'adresse e-mail.
- Authentification via JWT (JSON Web Token).
- Gestion des rôles (ex: candidat, recruteur).

Gestion du Profil Utilisateur

- Ajout, modification et consultation des informations personnelles.
- Ajout des sections suivantes dans le profil :
 - Éducation
 - Expériences professionnelles
 - Langues parlées
 - Certifications
 - Projets personnels ou académiques

Gestion des Fichiers

- Upload de fichiers (CV, photos, documents) vers static/uploads.
- Attribution d'un nom unique au fichier (UUID).
- Sauvegarde du chemin dans la base de données.
- Possibilité de consulter les fichiers uploadés.

Consultation et Filtrage des Offres

- Affichage de la liste des offres de stages.
- Filtrage des offres par :
 - Titre
 - Ville
 - Domaine
 - Type (Stage, Alternance...)
- Consultation détaillée d'une offre.

Interfaces Réalisées



Stage Connect



Hire Talent
Find Success

We are the best job
portal platform

This app will help you manage
your tasks easily.

Get started →



Next

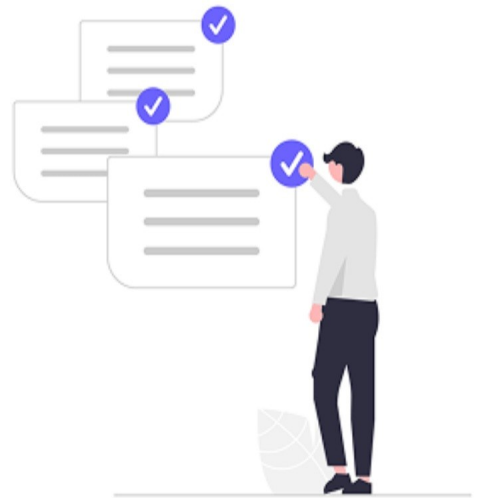


The place where works
find you

Group your tasks by categories
and stay focused.



Next



Let's start your career
with us now!

Track your progress and achieve
your goals.



Finish



Stage Connect



Email



Password



Remember Me

Sign In

Don't have an account? [Sign up](#)

[Forgot password?](#)

OR



Sign Up With Google



Stage Connect

Choose your profile type

Choose whether you are looking for a job
or you are an organization / company that
need employees



You are an intern

I want to find an internship



You are a recruiter

I want to find interns



You are a responsible

I am an intern's supervisor

Continue



What is your Expertise?

Please select the field of expertise (up to 3)

☐ Accounting and finance

☒ Engineering

☐ IT and Software

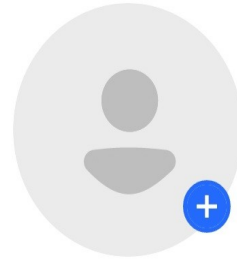
☐ Media and Design

☐ Sales and Marketing

☐ Management and consultancy

☐ Writing

Continue



First Name

Middle or Last Name

Date of Birth



Email



Phone Number



Gender



Continue




Good Morning!
BELLIL Mohamed

Search for a job or company

Recommendation

[See All](#)

 **UI/UX Designer**
Google LLC

San Francisco, CA
\$8,000 - \$20,000 / month

Full Time Remote Con

Recent jobs

[See All](#)

 **UI/UX Designer**
Google LLC

San Francisco, CA
\$8,000 - \$20,000 / month



Home



Saved jobs



Applications



Message




Profile




Search for a job or company



 **UI/UX Designer**
Google LLC


San Francisco, CA
\$8,000 - \$20,000 / month

Full Time Remote Con

 **Sales & Marketi...**
Paypal

New York, NY
\$7,000 - \$18,000 / month

Full Time Remote Con

 **Backend Devel...**
Amazon

Seattle, WA
\$9,000 - \$22,000 / month

Full Time Remote Con

 **DevOps Engine...**

← Saved jobs

🔍 Search for a job or company 🔍

 **UI/UX Designer**
Google LLC

San Francisco, CA

\$8,000 - \$20,000 / month

Full Time

Remote

Con

 **Sales & Marketi...**
Paypal

New York, NY

\$7,000 - \$18,000 / month

Full Time

Remote

Con

 **Backend Devel...**
Amazon



Home



Saved jobs



Applications



Message



Profile

← Saved jobs

🔍 Search for a job or company 🔍

 **UI/UX Designer**
Google LLC

San Francisco, CA

\$8,000 - \$20,000 / month

Full Time

Remote

Con

Remove from Saved ?

 **Sales & Marketi...**
Paypal

New York, NY

\$7,000 - \$18,000 / month

Full Time

Remote

Con

Cancel


Yes, Remove

← Applications


🔍 Search for a job or company 🔍

 **UI/UX Designer**
Google LLC

Application Sent

 **Sales & Marketing**
Paypal

Application Sent

 **Backend Developer**
Amazon

Application Rejected

 **DevOps Engineer**
Microsoft

Application Sent

 **iOS Developer**



Home



Saved jobs



Applications



Message



Profile

← Application Status



Machine Learning Engineer

Tesla

Palo Alto, CA

\$9,500 - \$23,000 / month

Full Time

Remote

Contract

Your Application Status

Application Accepted

Delete Application

← Profile



BELLIL Mohamed Nadir

Software Engineer



Contact Information



Summary



Work Experience



Education



Projects



Certifications



Internships



Skills



Languages



CV/Resume



Home



Saved jobs



Applications

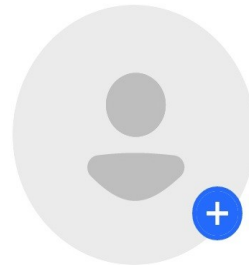


Message



Profile

← Profile Details



First Name

Mohamed Nadir

Middle or Last Name

BELLIL

Date of Birth

18/07/2002



Email

bellil.mohamedndir@gmail.cor



Phone Number

000000000000



Gender

Male



Current Position

Software Engineer



Save

← Contact Information



Address

570 route de ganges montpe

+33

Phone Number

000000000000



Email

bellil.mohamedndir@gmail.c

Save

← Summary

Summary (Max. 500 characters)

Summary

0 / 500

Save

← Education

Education

Education ▼

Course

Course

University

University

From

From ▼

To

To ▼

Graduated

☐

GPA

GPA ▼

Total

Total ▼

Description(Optional)

Description

0 / 200

Save

← Work Experience

Job Title

Job Title

Company

Company

From

From ▼

To

To ▼

I currently work here

☐

Description(Optional)

Description

0 / 200

Employment Type

Employment Type ▼

Location

Location

Job level

Job level ▼

Job Function

Job Function ▼

Save

← Certifications

Title

Title

Organization

Organization

Date of Issue

Date...

Expiration Date

Expi...

This Credential will not expire

☐

Credential ID (Optional)

Credential ID (Optional)

Credential URL (Optional)

Credential URL (Optional)

Save

← Skills

Skill

Skill



Java X

Kotlin X

Jetpack compose X

SQL X

Save

← Languages

Language

Language



Proficiency

Proficiency



Save

← CV/Resume

Upload CV/Resume



Browse File



cv.pdf

199690 Kb



Save

✕ Filter options

Location & Salary

 Search for a job or company

30000€ - 120000€



Monthly/Yearly
per month



Work Type



- ☐ Onsite (Work From Office)
- ☐ Remote (Work From Home)

Job Level



- ☐ Internship
- ☐ Entry Level / Junior, Apprentice
- ☐ Associate / Supervisor
- ☐ Mid- Senior Level
- ☐ Director / Executive

Employment Type



- ☐ Full Time
- ☐ Part Time
- ☐ Freelance
- ☐ Contractual

Experience



- ☐ No Experience
- ☐ 1 - 5 Years
- ☐ 6 - 10 Years
- ☐ More than 10 Years

Education



- ☐ Less Than High School
- ☐ High School
- ☐ Diploma
- ☐ Bachelor's Degree
- ☐ Master's Degree
- ☐ Doctoral or Professional Degree

Job Function



- ☐ Accounting and Finance
- ☐ Administration
- ☐ Architecture and Engineering
- ☐ Arts and Sports
- ☐ Customer Service
- ☐ Education and Training
- ☐ General Services
- ☐ Health and Medical
- ☐ Hospitality and Tourism
- ☐ Human Resources
- ☐ IT and Software
- ☐ Legal
- ☐ Management and Consultancy
- ☐ Manufacturing and production
- ☐ Media and Creatives
- ☐ Public Services and NGOs
- ☐ Safety and Security
- ☐ Sales and Marketing
- ☐ Sciences
- ☐ Supply Chain
- ☐ Writing and Content

Reset

Apply

Avancement et prochaines étapes

Ce qui est réalisé

- Mise en place de la structure Clean Architecture côté mobile.
- Écrans principaux de l'application (connexion, inscription, affichage des offres).
- Backend initial avec gestion des utilisateurs, offres et authentification sécurisée.
- Communication mobile-backend opérationnelle via API REST sécurisée par JWT.

Ce qui reste à faire

- **Partie entreprise** : profils, la gestion des candidatures, messagerie intégrée.
- **Partie université** : Profil, validation académique des stages, gestion des conventions.
- **Fonctionnalités avancées** : notifications automatiques, génération de PDF

Modules complémentaires

suivi du stage, rapports périodiques, vidéo de présentation des entreprises.